I. INTRODUCTION

In this report we describe an adaptive error control system to be used on the ATS-1 VHF satellite link for remote-terminal to computer and computer to computer communication. Previous studies on this channel using a rate 1/2 convolutional code for error correction showed[1] that significant improvement can be made in error rates. However, after examining the time varying nature of the errors and the fact that the convolutional code worked even under worst case conditions, it was felt that a comparable improvement in error rates could be obtained using an adaptive scheme for error control. Such a scheme would not employ a fixed overhead for correction. Rather, the overhead would be determined by channel conditions, so that the average information rate is smaller than that given by any fixed scheme for the same improvement in error probability.

For the sake of practicability, the error control scheme would have to fit the existing structure of THE ALOHA SYSTEM without involving major changes in network protocols or equipment. The system described here satisfies this property since it is a natural extension of the automatic-repeat-request (ARQ) system currently used. Further, no changes are needed except at the receiving end, where a program controlled error correction box would be added. The basically insensitive nature of ARQ to changes in channel conditions is retained, opening the possibility of using a similar scheme for channels having very different error characteristics.

II. MOTIVATION FOR EXTENDED ARQ

When digital data is to be transmitted from one point to another in a noisy environment, some sort of error control mechanism is usually required to keep error rates below a tolerable level. There are two basically different classes of techniques which are used for this purpose. The first one, called forward error correction (FEC), involves encoding messages into longer codewords at the transmitter such that the redundant part can be used at the receiver to determine the pattern of errors that occurred. The decoder (Fig. 1) computes the redundant bits from the received message and compares them to the received redundant bits. The difference contains information about the error pattern that permits the receiver to correct the errors. The reverse channel shown in Fig. 1 is not used in FEC.

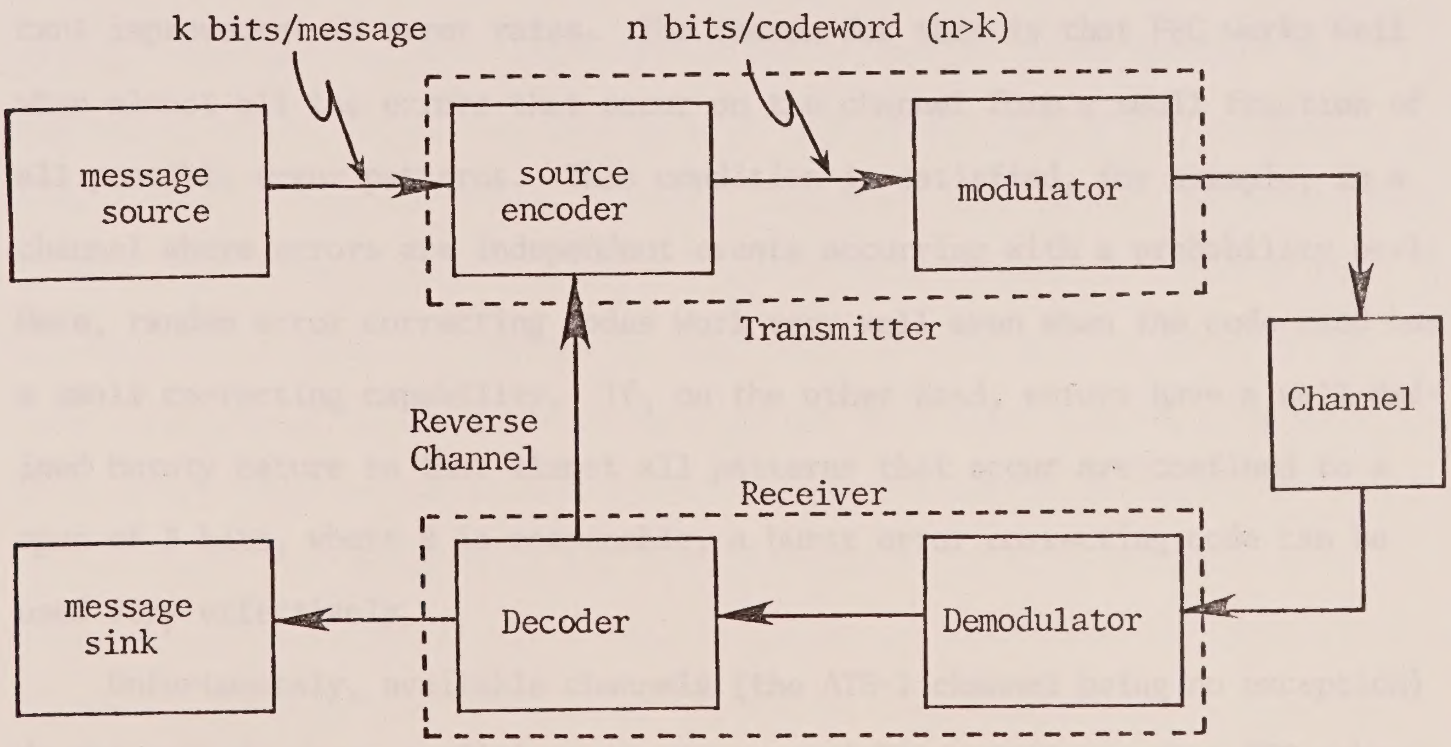


Figure 1. A General Error Control System. (The reverse channel is not used when the system is in FEC mode.)

The other class of techniques is known as Automatic-Repeat-Request (ARQ). Here, redundancy is added to messages as in FEC, but the decoder only tries to determine if errors have occurred, making no attempt to identify the pattern. If errors have occurred, the receiver uses a reverse channel (Fig. 1) to signal a negative acknowledgement (NACK) to the transmitter, causing it to retransmit the message. If the receiver determines that no errors have occurred, it sends a POSACK which inhibits the retransmission. In one variation of ARQ, used on THE ALOHA SYSTEM, only POSACKS are sent, the message being retransmitted automatically after a time if no POSACK has been received. All ARQ systems require the message to be bufferred at the transmitter till a POSACK is received.

It has been found[2] that for many real digital data communication channels, high rate FEC schemes that can be simply implemented do not result in significant improvement in error rates. The reason for this is that FEC works well when almost all the errors that occur on the channel form a small fraction of all possible error patterns. This condition is satisfied, for example, in a channel where errors are independent events occurring with a probability $p<<1$. Here, random error correcting codes work very well even when the code used has a small correcting capability. If, on the other hand, errors have a well-defined bursty nature so that almost all patterns that occur are confined to a span of B bits, where B is reasonable, a burst error correcting code can be used very effectively.

Unfortunately, available channels (the ATS-1 channel being no exception) do not satisfy these conditions to any appreciable extent, so that FEC schemes that implement easily do not work well. Coding schemes have been developed

The other class of techniques is known as Automatic Repeat Request (ARQ). Here, redundancy is added to messages as in FEC, but the decoder only tries to determine if errors have occurred, making no attempt to identify the pattern. If errors have occurred, the receiver uses a reverse channel (Fig. 1) to signal a negative acknowledgment (NACK) to the transmitter, causing it to retransmit the message. If the receiver determines that no errors have occurred, it sends a POSACK which confirms the correct reception. In one variation of ARQ, used in [illegible], only POSACKS are sent, the message being retransmitted automatically after a time if no POSACK has been received. All ARQ systems require the message to be retained at the transmitter till a POSACK is received.

It has been found[2] that for many real digital data communication channels, high rate FEC codes that can be cheaply implemented do not result in significant improvement in error rates. The reason for this is that FEC works well when almost all the errors that occur in the channel form a small fraction of all possible error patterns. This condition is satisfied, for example, on a channel where errors are independent events occurring with probability $p \ll 1$. Here, random error correcting codes work very well even when the code used has a small correcting capability. If, on the other hand, errors have a well-defined bursty nature so that almost all patterns that occur are confined to a span of b bits, where b is reasonable, a burst error correcting code can be used very effectively.

Unfortunately, practical channels (the ATS-1 channel being an exception) do not satisfy these conditions to any appreciable extent, so that FEC schemes that implement easily do not work well. Coding schemes have been developed for channels with composite burst and random errors, but here either the decoding algorithm is complicated or the rate of the code is low.

ARQ schemes on the other hand are much simpler to implement since only error detection needs to be done. The probability of error in ARQ is determined by the probability of an error going undetected, and this number is of the order of 2^{-p}, where p is the number of parity bits, independent of the nature of errors on the channel. This insensitivity of ARQ makes it very attractive for use on channels with time dependent error characteristics.

Improvements on ARQ

In spite of their simplicity and inherent insensitivity, pure ARQ schemes suffer from inefficient use of the channel especially when multiple retransmits are required for a particular message. Hybrid error control schemes which have an FEC capability embedded inside ARQ get around this problem to some extent. The question that naturally arises is whether it is possible to improve on hybrid ARQ-FEC by cutting down the average number of retransmits even further. This can indeed be done. The basic idea behind the improvement is that discarding a packet which is found to have errors, is throwing away valuable information -- information that could well be used in conjunction with a retransmit to correct many more errors than would otherwise be possible. If decoding based on two copies fails, another retransmit is called for and correction is attempted using the cumulative store of three copies. This process continues till either a correct packet is received or it is possible to decode the message from the copies. It can be seen heuristically, that the number of retransmits required would be reduced considerably if the redundancy

present in the form of message multiplicity is carefully used. Such a system will work much better with a time varying channel than either FEC, pure ARQ or hybrid ARQ-FEC. Consider, for example, what happens when each of these schemes is used on a channel which periodically enters a 'noisy mode'. For such a channel, errors are very likely to occur in the retransmit during the noisy mode. FEC will fail completely if the errors are bad enough. ARQ will keep requesting retransmits, finally giving up when too many retransmits have occurred. Hybrid ARQ-FEC will not work any better unless the inbuilt FEC is able to correct at least one of the copies. The extended ARQ scheme described above has a better chance of being able to correct such errors because of the large redundancy present.

In the following sections, we discuss the nature of errors on the given channel and describe the extended ARQ scheme more completely, giving an algorithm for decoding with two copies.

III. CHANNEL STATISTICS

Like most data communication channels, the ALOHA ATS-1 channel is characterized by a mix of random error and burst errors; also, as is true for many such channels, error patterns are such that with high rate codes forward error correction does not improve error rates significantly.

Random errors on this channel may be attributed to Gaussian noise, short duration impulse noise that affects only one or two bits and discriminator noise during periods of low signal. Most of these errors occur in the form of a pair of adjacent errors due to the differential encoders used.

Burst errors are caused by satellite co-channel interference and auto-ignition noise at the frequency of operation. The latter occurs in the form of high energy, short duration pulses which cause the receiver to lock up in one or the other state for several bit times, resulting in an error burst. Loss of bit synchronization at the receiver also causes bursts, but these are typically much longer and tend to occur towards the tail of the message.

When the distribution of errors per packet is examined (Fig. 3), it is noticed that double adjacent errors account for around 40% of all packets in error. The long tail of this curve is the single factor which prevents forward error correction from working well on this channel. Consider the improvement that results when a random error correcting code with correcting ability t is used. From the cumulative figures in Fig. 3, 40% of the e-rors remain uncorrected when t=8, 33% remains uncorrected with t=12, and 29% with t=16. It is clear that the largest gains are to be obtained by correcting just two or four errors. Any further improvement using FEC is marginal as long as we

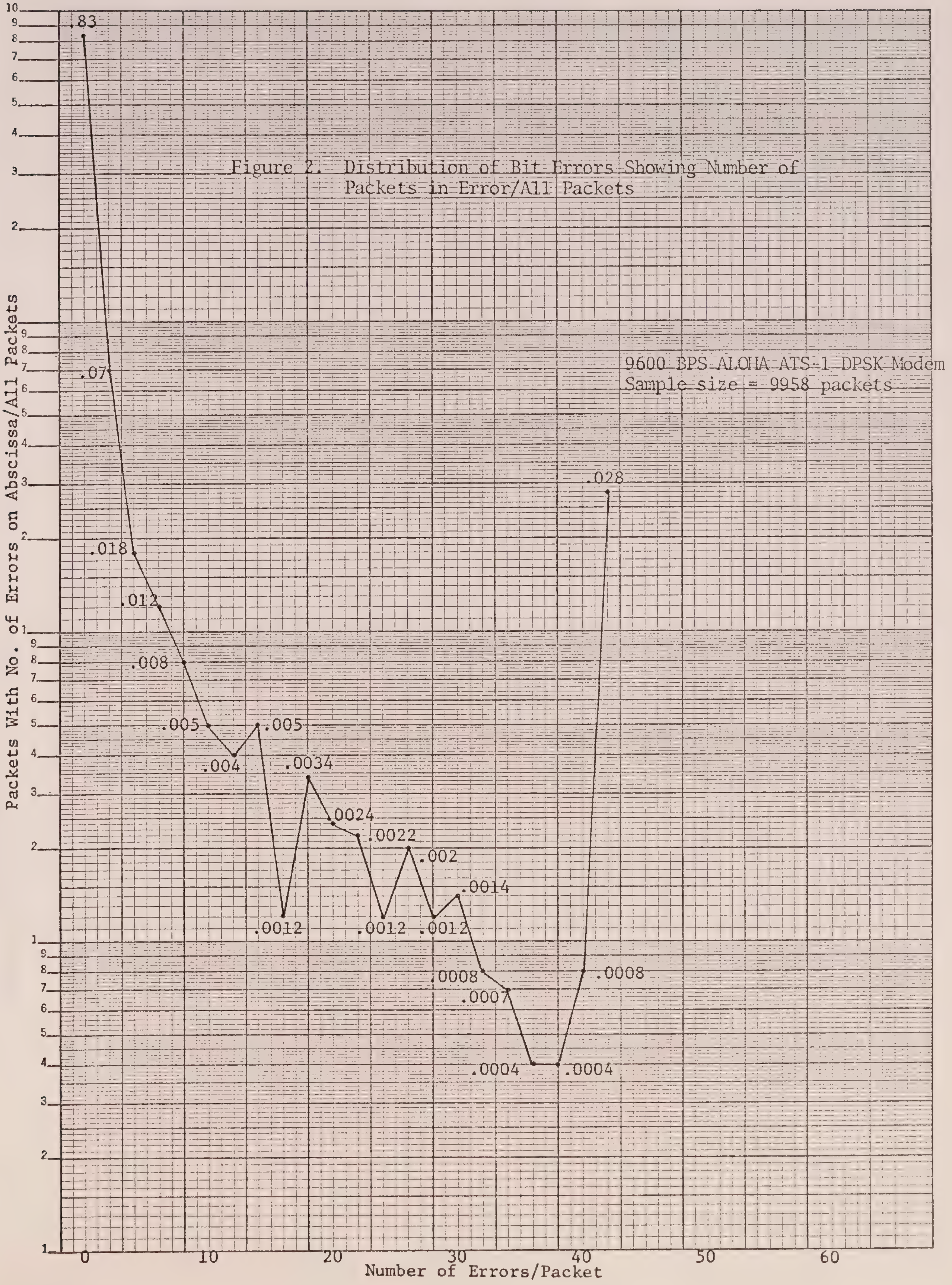
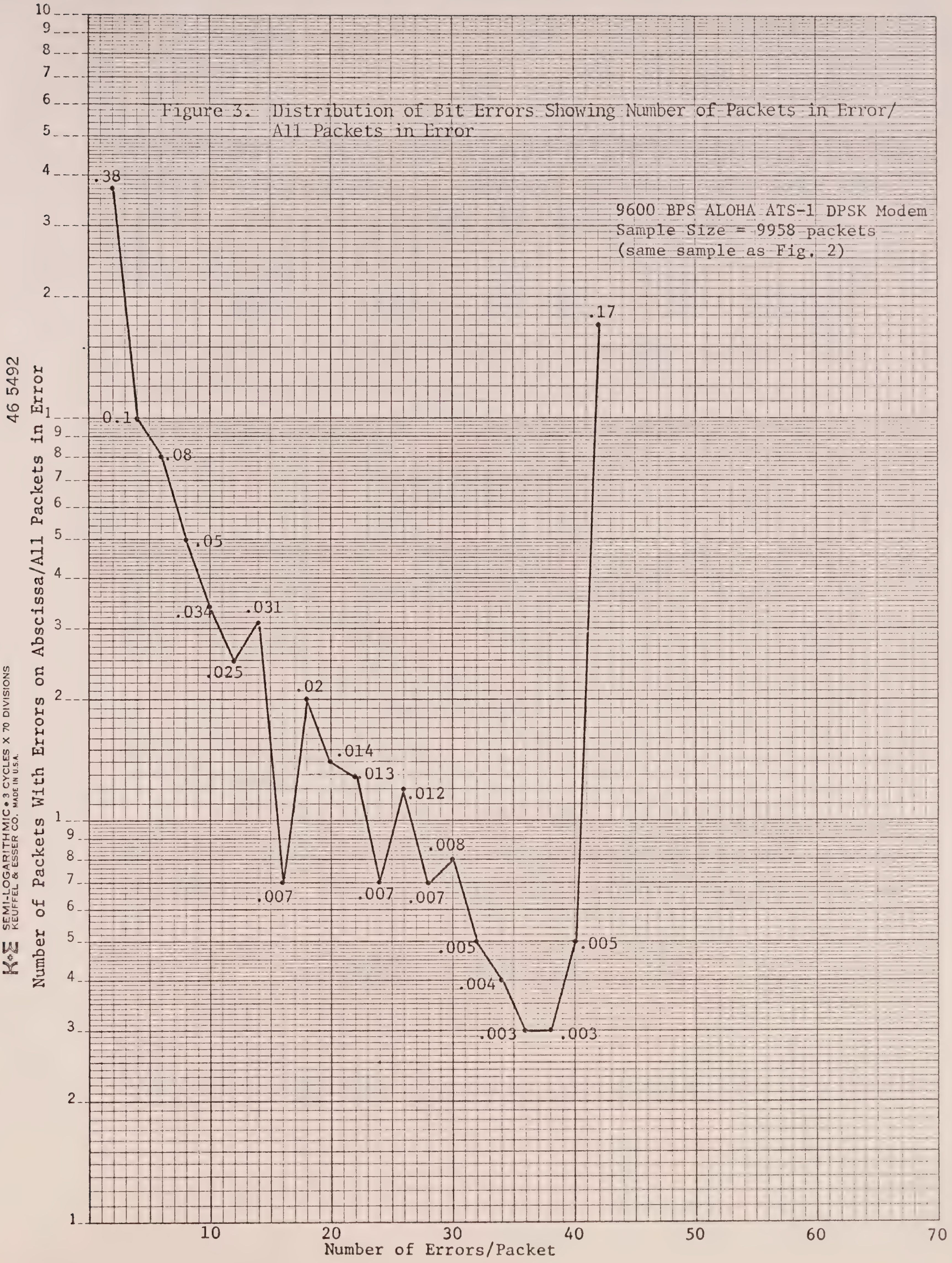


Figure 2. Distribution of Bit Errors Showing Number of Packets in Error/All Packets

Figure 3. Distribution of Bit Errors Showing Number of Packets in Error/ All Packets in Error



restrict ourselves to high rate codes and simple implementation. If correction based on repetition is used, it is estimated that with a single repeat around 40 errors occurring in one of the packets can be corrected, accounting for all but 17% of the packets in error.

Since significant improvement results from correcting double adjacent errors using FEC, a hybrid system which uses repetition correction with an embedded FEC will work better than a pure repetition system. We call this scheme extended hybrid ARQ-FEC. The throughput T defined as

$$T = \frac{\text{Number of Message Bits}}{\text{Number of Channel Bits}}$$

will now be calculated for this scheme.

Assume a message length of ∿700 bits and a parity of 24 bits* for the embedded FEC. The throughput will be given by

$$T = \frac{n_m}{(n_m+n_p)+f(n_m+n_p)}$$

where n_m = number of message bits
n_p = number of parity bits
f = fraction of all packets uncorrected by the embedded FEC

This formula assumes that all errors can be corrected with a single repetition, an assumption that introduces only second order effects. From Fig. 2 we have $f \approx 0.1$

$$\therefore T = \frac{700}{724+72\cdot 4} \simeq \frac{7}{8} \simeq 0\cdot 88.$$

*Correction of single paired errors while maintaining a good detection capability, for ARQ to work well, will require around 24 bits.

For comparison, consider the correcting capability of a pure FEC system that gives the same throughput. The best known constructive codes, the BCH codes give an error correction rate of 0.015[3], which is around 11 bits out of 700. For t=12, 33% of the errors remain uncorrected (previous discussion).

The arguments presented above give the rationale for choosing the extended hybrid ARQ-FEC scheme. No claim is made here about the optimality of correction by repeating the message verbation. It may be argued, for example, that this is an inefficient way of using the repeat packet and that more powerful methods exist where the repeat packet would be some encoded version of the first packet. Such schemes would probably perform better, but whether the added complexity both in decoding and at the transmitter is worth the gains is an open question.

IV. SYSTEM DESCRIPTION

Here we give a detailed description of the extended hybrid ARQ-FEC system discussed ear4ier. This system uses repetition of data along with embedded FEC to correct errors. Each packet carries sufficient redundancy to enable simple error patterns to be corrected. If an error packet that cannot be corrected by the FEC is encountered, a repeat of the information is initiated and correction will be attempted using the cumulative store of information.

Figure 4 shows a schematic of the error control unit at the receiver.

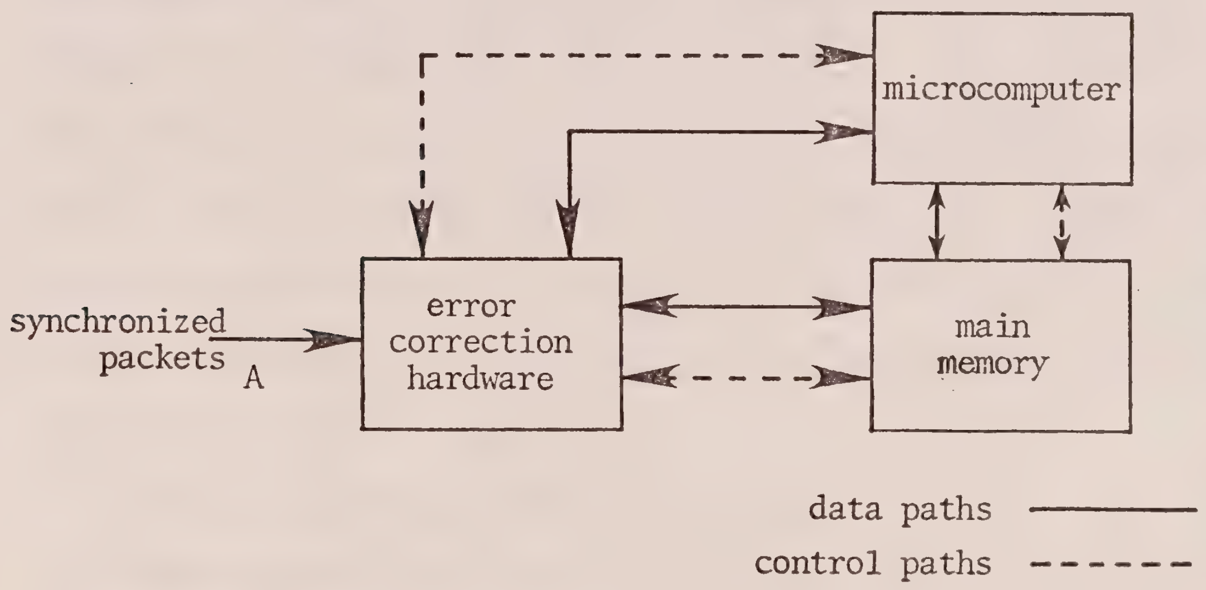


Figure 4. Error Control Unit

Synchronized* packets which are determined to have the appropriate identification are input to the error correction base at point A. This hardware base, under loose control of the microcomputer calculates the syndrome of the received

*The problem of packet synchronization for the ALOHA ATS-1 channel is considered in another report.[4]

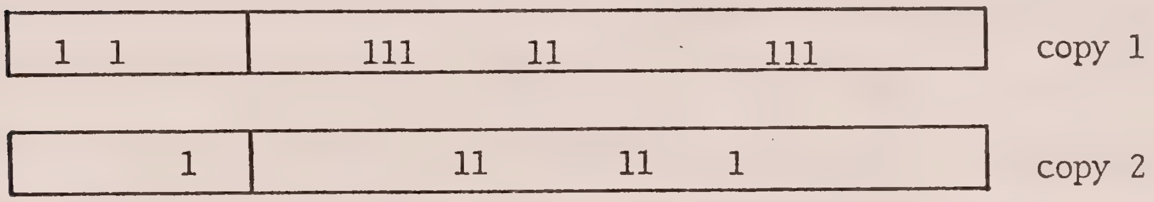
word. If the syndrome is zero, it is assumed that no errors have occurred and an ACK is sent to the transmitter to indicate that a packet has been correctly received. If the syndrome is non-zero the microcomputer initiates a burst trapping procedure which determines if the syndrome corresponds to an error burst of length two or less. If such an error pattern is found, correction is done and an ACK is sent. If the errors are uncorrectable, the packet is saved in the computers main storage, and the system waits for the repeat packet. On arrival, this packet is checked for a correctable error pattern exactly as before. If it also has uncorrectable errors, a discrepancy packet is formed by taking the bit-by-bit exclusive-or of the buffered packet and the new packet. If the weight of this packet is more than a predetermined number, it is assumed that the two packets were not identical to start with, and that one of them is the result of a false alarm by the synchronization unit. The receiver waits for the genuine copy to be received. Once two closely matching copies are available a decoding procedure is started which tries to determine the correct message from the copies. The decoding algorithm will be given in the next section.

The idea of using bufferred information to correct errors can be extended to where decoding operates on three or more copies. The potential correcting ability evidently grows rapidly with the number of copies but so does the complexity of decoding. For this system, we will consider decoding with just two copies since the expected improvement is adequate and the extra complexity is not justified. The complete operation is summarized by the state diagram of Fig. 5.

V. TWO-COPY DECODING

This section gives a decoding algorithm which can be used to determine the transmitted packet when there are two copies of a packet and each copy has some errors. The errors in each packet will moreover be of the type which are not correctable by the embedded FEC capability since two-copy decoding is invoked only when FEC fails on both packets. The basic idea is presented first and the actual algorithm is given later. A qualitative idea of how this algorithm would perform is also given.

Consider two copies of a packet having the error patterns shown below:



As long as errors in the two copies do not occur in corresponding columns all the information about error positions is recoverable and any error pattern can be corrected.

Let $r_1(x)$ and $r_2(x)$ denote the two received copies of a transmitted packet $v(x)$, and $e_1(x)$ and $e_2(x)$ denote the error patterns in $r_1(x)$ and $r_2(x)$. Define $d(x)$ to be the modulo-two sum of $r_1(x)$ and $r_2(x)$.

$$\begin{aligned} d(x) &= r_1(x) + r_2(x) \\ &= v(x) + e_1(x) + v(x) + e_2(x) \\ &= e_1(x) + e_2(x) \end{aligned}$$

Since errors in $r_1(x)$ and $r_2(x)$ are assumed to be disjoint, $d(x)$ contains positional information about all the errors in these two packets. Determining the error pattern therefore corresponds to making a number of binary decisions about the belonging of non-zero bits in $d(x)$ to $r_1(x)$ or $r_2(x)$. If k_1 errors occurred in $r_1(x)$ and k_2 in $r_2(x)$, there are $2^{k_1+k_2}$ ways in which the k_1+k_2 non zero bits in $d(x)$ can be assigned to $r_1(x)$ and $r_2(x)$. One of these assignments must correspond to the actual error patterns $e_1(x)$ and $e_2(x)$. Whether a given assignment is the right one will be determined by calculating the syndrome of $r_1(x)$ modified by adding the assigned bits, and checking if the syndrome is zero. This procedure can be modified to correct some of the overlapping error patterns also. If the pattern of overlaps can be confined to a single burst $\leq b$, where b is the burst correcting ability of the embedded code, then it can be corrected. The syndrome of $r_1(x)$ modified by each assignment is checked to see if it corresponds to a correctable pattern.

In practice, k_1+k_2 will be large-typically anywhere from 4 to 40 when the channel is noisy. The bit assignment method described above cannot be practically implemented. When k_1+k_2 is greater than 10 or so. Advantage can be taken of the fact that errors on this channel are not completely independent, but tend to occur in bursts. To give an idea of the correlated nature of errors, for a packet having around 30 bit errors, the errors can be grouped into three or four bursts with a guard band of just 5. This suggests that an assignment procedure that works with bursts instead of bits will be practical.

Here, the nonzero bits in $d(x)$ are divided into a number of bursts $b_1(x), b_2(x) \ldots b_\ell(x)$ with guard band g. By this we mean that each $b_i(x)$ starts and ends with a 1, and does not have g or more consecutive zeros. Further, each $b_i(x)$ is separated from its neighbors by at least g consecutive zeros. We now make the assumption that a burst in $d(x)$ contains errors either from $r_1(x)$ or $r_2(x)$ but not both. This condition being satisfied, one of the 2^ℓ possible patterns of assigning ℓ bursts in $d(x)$ to $r_1(x)$ and $r_2(x)$ must correct the errors in $r_1(x)$ and $r_2(x)$. The reduction in computation is not obtained free of cost since the number of correctable error patterns is reduced somewhat. There is clearly a tradeoff here between computation complexity and error correcting ability, and the tradeoff is determined by the guard band g. Increasing g will lower the number of bursts in the discrepancy packet, and hence the number of assignments to be made. But since resulting bursts are longer, there is a greater chance that a burst in $d(x)$ will contain errors from both $r_1(x)$ and $r_2(x)$. It is interesting to note that with decreasing number of bursts, computation complexity reduces exponentially whereas the number of uncorrectable errors is not affected as drastically.

The optimum value of g for the channel can be determined as follows: the maximum number of bursts that can be handled is first determined from time constraint on the calculation. Let this number be ℓ*. Channel test data with known error patterns is then run through a program which counts the number of bursts in discrepancy packets. Several runs are made, each time with a

---

*ℓ is expected to be 8 or 9. This gives 256 or 512 assignments depending on the value chosen.

different value of g. The distribution of bursts per discrepancy packet is then plotted for different g. The smallest value of g is chosen for which a distribution has say 95% of the packets with $\leq \ell$ bursts.

Correction Algorithm

The correction procedure basically involves taking all the possible 2^ℓ combinations of ℓ bursts in $d(x)$, adding each combination to $r_1(x)$ and calculating the syndrome of each sum. The first syndrome which is zero corresponds to the correct combination. If each of the 2^ℓ syndromes is calculated using a feedback shift register, the calculation time will be large since each calculation involves $\sim$700 shifts.

Actually only ℓ linearly independent syndromes need be calculated using the feedback circuit and the others can be generated by linear combinations of these ℓ syndromes.

Let $B_1 b_1(x)+B_2 b_2(x)+\ldots B_\ell b_\ell(x)$ denote a particular combination of bursts. A given B_i will be non-zero only if the corresponding $b_i(x)$ appears in the combination. There is a 1:1 correspondence between the binary number $0,1,2\ldots 2^\ell-1$ and burst combinations $B_0, B_1, B_2, \ldots B_{2^\ell-1}$. For example, $B_0=0$, $B_4=0.b_1(x)+0.b_2(x)+1.b_3(x)$ and $B_{2^\ell-1} = b_1(x)+b_2(x)+\ldots b_\ell(x)$.

Let $\rho_i(x)$ be the syndrome corresponding to $b_i(x)$, and let the ℓ independent syndromes chosen be:

$$s_1(x) \triangleq \rho_1(x)$$
$$s_3(x) \triangleq \rho_1(x) + \rho_2(x)$$
$$\vdots$$
$$s_{2^\ell-1}(x) \triangleq \rho_1(x) + \rho_2(x) + \ldots + \rho_\ell(x)$$

These are obviously the syndromes corresponding to the ℓ burst combinations $B_{2^{k-1}}$, $k=1,2,\ldots\ell$. All the 2^{ℓ} syndromes can be generated by the following recursive formula:

$$s_0(x) \triangleq 0$$
$$s_1(x) \triangleq \rho_1(x) = s_0(x) + s_1(x)$$
$$s_2(x) \triangleq \rho_2(x) = \rho_1(x) + \rho_1(x) + \rho_2(x) = s_1(x) + s_3(x)$$
$$s_3(x) \triangleq \rho_3(x) = \rho_2(x) + \rho_1(x) = s_2(x) + s_1(x)$$
$$\vdots$$
$$s_j(x) \triangleq \rho_j(x) = s_{j-1}(x) + s_q(x)$$

where $q = (j)_{\text{base }2} \oplus (j-1)_{\text{base }2}$

For example if $j=5$,

$$q = 101 \oplus 100 = 011 = 1$$

$s_j(x)$ is generated from $s_{j-1}(x)$ by adding one of the syndromes from the linearly independent set. This procedure has the advantage that the syndromes are generated in the correct sequence, $s_0(x), s_1(x), s_2(x), \ldots s_{2^{\ell}-1}(x)$, using just modulo-2 additions.

The syndrome corresponding to $r_1(x)+B_i$ can be determined simply as $S_1(x)+s_i(x)$ where $S_1(x)$ is the worst case the complete error correction will require calculation of ℓ syndromes and 2^{ℓ} modulo-2 additions, with a test for zero after each addition.

APPENDIX A

DISTRIBUTION OF ERRORS IN GROUPS OF 16 AND 32 BITS

Groups of 16 Bits: Number of groups out of 100 having less than or equal to N Errors/Group
Sample Size = 37,4320 groups

Channel* Condition	Numbers of Errors (N) → 0	≤1	≤2	≤3	≤4	≤5	≤6	≤7	≤8	≤9	≤10	≤11
I	995	995.4	997.9	998.1	998.7	998.9	999.2	999.4	999.7	999.8	999.9	999.9
II	840	840.4	842.9	851.9	861.9	876.2	906.2	937.2	964.2	983.2	991.9	995.8
III	840	850.7	959.7	965.5	981.5	985.1	991.6	996.1	999.0	999.5	...	...
Total of I,II,III	964	965.2	976.3	977.9	980.8	982.9	987.2	992.1	995.7	998.1	999.1	999.6

Groups of 32 Bits: Number of groups out of 1000 having less than or equal to N Errors/Group
Sample Size = 18,7160 groups

Channel* Condition	Number of Errors (N) → 0	≤1	≤2	≤3	≤4	≤5	≤6	≤7	≤8	≤9	≤10	≤11	≤12
I	992.2	992.6	997.0	997.07	997.8	997.9	998.2	999.1	999.4	999.5	999.2	999.8	999.99
II	840	840.3	844.2	844.2	844.8	844.9	845.5	846.4	849.0	855.0	864.6	878.8	895.8
III	750	760	908	914	962.8	965.4	973	974.7	981.7	985.2	988.2	994.9	1000...
Total of I,II,III	954.8	955.9	971.9	972.5	977.0	977.3	978.2	978.5	979.6	980.7	982.2	984.5	989.6...

*Note: The data above represents sample runs taken on the channel at different times. There are approximately 10 runs involved here. Since the channel is time varying, the channel condition for each run was classified into one of three categories:

I - Channel in "good" phase, relatively few errors

II - Channel in bad phase, error distribution has long tail

III - Channel in bad phase, but tail of distribution is not long